La sémantique sans variable

Nicolas Guilliot

2 juin 2006

Basé sur Jacobson (1999) et le cours *Binding and Ellipsis* donné par Heim et Jacobson à LSA 2005.

Le point de départ

• théorie basée sur la grammaire catégorielle et sur l'idée qu'un modèle représentationnel tel que LF n'est pas nécessaire pour le calcul de l'interprétation : calcul direct et très local de l'interprétation ;
• rejet de la notion de variable comme un objet théorique : idée que 2 variables différentes (x_1 et x_2) ne vont créer aucune différence sémantique (John loves x_1 = John loves x_2), or 2 objets théoriques distincts d'un modèle devraient aboutir à 2 interprétations distinctes ;
• les pronoms ne sont pas des variables non plus : ils expriment plutôt la fonction d'identité ($\lambda x.x$).

1 Les bases formelles

Modèle basé sur la syntaxe et la sémantique formelles :
- syntaxe ⇒ grammaire catégorielle et types syntaxiques ;
- sémantique ⇒ sémantique formelle vériconditionnelle et types sémantiques.

1.1 sémantique formelle et types sémantiques

Voir Heim and Kratzer (1998) pour plus de détails.

La sémantique vériconditionnelle (et le calcul du sens) obéit à une théorie des types : idée que tous les éléments de la langue ne vont pas dénoter la même chose.

Distinction entre termes singuliers, prédicats, et propositions (entre autres).

Les entités individuelles → type e (pour entity).

(1) $[\![Jean]\!]$ = l'individu 'Jean' dans le monde, $\in D_e$

Les propositions → type t.

(2) [[Jean regarde la télé]]=1 ssi Jean regarde la télé, sinon 0, $\in D_t$

Les prédicats → type $\langle e, t\rangle$ si 1 argument, type $\langle e, \langle e, t\rangle\rangle$ si 2 arguments, ...

(3) $[\![regarder]\!] = \lambda y \in D_e[\lambda x \in D_e.x \textit{ regarde } y]$, $\in D_{\langle e,\langle e,t\rangle\rangle}$

(4) $[\![donner]\!] = \lambda z \in D_e[\lambda y \in D_e[\lambda x \in D_e.x \textit{ donne } z \text{ à } y]]$, $\in D_{\langle e,\langle e,\langle e,t\rangle\rangle\rangle}$

Les déterminants → type $\langle\langle e, t\rangle, e\rangle$.

1.2 grammaire catégorielle et types syntaxiques

Jacobson (1999) utilise une distinction similaire en syntaxe, en fonction de la catégorie syntaxique des unités syntaxiques. Cette notion de type syntaxique constitue la base des grammaires catégorielles (GC), dans lesquelles chaque unité a une catégorie syntaxique qui rend compte de son contexte d'occurrence.

Les entités individuelles → catégorie NP.

(5) $\dfrac{Jean}{NP}$

Les propositions → catégorie S.

(6) $\dfrac{Jean\ regarde\ Marie}{S}$

Les prédicats → catégorie $S/_L NP$ si 1 argument, catégorie $(S/_L NP)/_R NP$ si 2 arguments[1], ...

(7) $\dfrac{partir}{S/_L NP}, \dfrac{regarder}{(S/_L NP)/_R NP}$

Les noms communs → catégorie N.

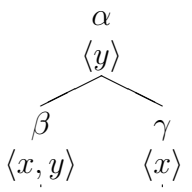
(8) $\dfrac{chat}{N}$

Les déterminants → catégorie $NP/_R N$.

1.3 Combinaison sémantique : application fonctionnelle

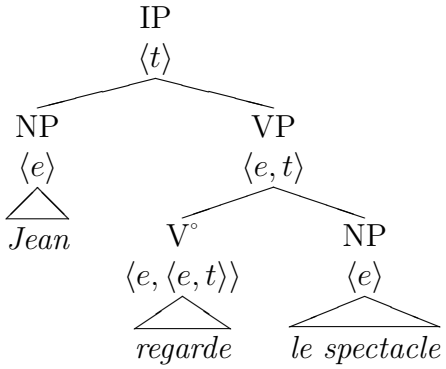
Le calcul sémantique dans la théorie standard repose essentiellement sur la règle d'application fonctionnelle :

(9) *Application fonctionnelle :*
Si α est un noeud branchant, β et γ ses filles, et $[\![\beta]\!]$ est une fonction dont le domaine contient $[\![\gamma]\!]$, alors $[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$.



La théorie des types et cette règle d'application déterminent les combinaisons possibles entre prédicats, entités individuelles et propositions :

(10)



[1]Se lit de la manière suivante : quelque chose qui prend un NP sur sa droite (R) pour renvoyer $S/_L NP$, ce résultat prenant un NP sur sa gauche (L) pour renvoyer S. Il existe d'autres notations en GC parfois : NP\S/NP.

Un prédicat de type $\langle e, \langle e, t\rangle\rangle$ comme *regarder* va se combiner avec une première entité individuelle $\langle e\rangle$ pour former l'interprétation du VP (1ère application fonctionnelle), puis avec une seconde entité individuelle (2ème application fonctionnelle) pour donner une valeur de vérité :

(11) ⟦Jean regarde le spectacle⟧=1
ssi ⟦regarde le spectacle⟧(⟦*Jean*⟧)=1
ssi ⟦regarde⟧(⟦le spectacle⟧)(⟦*Jean*⟧)=1
ssi ⟦regarde⟧(l'unique z : z est un spectacle)(Jean)=1
ssi $\lambda y \in D_e[\lambda x \in D_e.x\ regarde\ y]$(l'unique z : z est un spectacle)(Jean)=1
ssi Jean regarde le spectacle.

1.4 la combinaison syntaxique

Là encore, c'est le type qui détermine les combinaisons possibles :

(12)
$$\dfrac{\dfrac{Jean}{NP} \qquad \dfrac{\dfrac{regarde}{(S/_L NP)/_R NP} \qquad \dfrac{Marie}{NP}}{S/_L NP}}{S}$$

Les catégories syntaxiques retreignent aussi la combinaison. Le type $(\mathrm{S}/_L\mathrm{NP})/_R\mathrm{NP}$ autorise la combinaison seulement avec un NP sur sa droite (R) et renvoye $\mathrm{S}/_L\mathrm{NP}$, ce dernier autorisant la combinaison avec un NP sur sa gauche (L) pour renvoyer S.

1.5 type-shifting (sémantique et syntaxique)

Opération qui permet d'inverser la relation prédicat-argument. Le type de l'argument est modifié ('lifted') pour devenir lui-même le prédicat grâce à l'opérateur l :

(13) Sémantique : si α est de type $\langle a\rangle$, alors $[\![l_b(\alpha)]\!] = \lambda F_{\langle a,b\rangle}.F([\![\alpha]\!])$
Syntaxe : $l_B(\mathrm{A}) = \mathrm{B}/_R(\mathrm{B}/_L\mathrm{A})$

(14) *John left.*
Sémantique : ⟦$l(John)\ left$⟧=1
ssi $[\lambda P_{\langle e,t\rangle}.P(John)]([\![left]\!])$
ssi $[\lambda P_{\langle e,t\rangle}.P(John)](\lambda y.y\ left)$
ssi $[\lambda y.y\ \ left](John)$
ssi $John\ left$

Syntaxe :
$$\dfrac{\dfrac{l(John)}{S/_R(S/_L NP)} \qquad \dfrac{left}{S/_L NP}}{S}$$

Logiquement, *left* de type $\mathrm{S}/_L\mathrm{NP}$ et $\langle e, t\rangle$ prend comme argument *John* de type NP et $\langle e\rangle$. L'opérateur inverse la relation en syntaxe et en sémantique : le prédicat *left* devient l'argument.

1.6 Théorie standard des pronoms et variables

En sémantique formelle traditionnelle, chaque variable ou pronom dépend de la fonction d'assignation. Une variable dénote un individu, mais seulement relatif à un choix d'assignation de valeur. Cette fonction d'assignation est nourrie par les indices référentiels des entités individuelles (tout ce qui est de type e).

(15) Jean_1 aime le livre que$_3$ Marie_2 lit t_3.

La valeur de vérité de (15) va dépendre de la fonction d'assignation g qui associe 1 à $Jean$, 2 à $Marie$ et 3 à x (via l'abstraction lambda, non-détaillée ici) :

(16) $[\![Jean_1\ aime\ le\ livre\ que_3\ Marie_2\ lit\ t_3]\!]^g$=1
ssi $[\![aime]\!]^g([\![le\ livre\ que_3\ Marie_2\ lit\ t_3]\!]^g)([\![Jean]\!]^g)$=1
ssi $[\![aime]\!]$(l'unique x.Marie lit x, un livre)($[\![Jean]\!]$)=1
ssi $[\lambda y.\lambda z.z$ aime $y]$(l'unique x.Marie lit x, un livre)(Jean)=1
ssi Jean aime l'unique x.Marie lit x, un livre.

L'interprétation des pronoms va elle aussi dépendre exclusivement de la fonction d'assignation. Ainsi, un pronom va renvoyer à un individu via son indexation, et plus précisément la fonction d'assignation :

(17) $[\![il_1]\!]^{\left[\substack{1\rightarrow \textsc{Jean} \\ 2\rightarrow \textsc{Paul}}\right]}$ = Jean

2 La sémantique sans variable

• rejet de la notion de variable comme un objet théorique : idée que 2 variables différentes (x_1 et x_2) ne vont créer aucune différence sémantique (John loves x_1 = John loves x_2), or 2 objets théoriques distincts d'un modèle devraient aboutir à 2 interprétations distinctes ;
• les pronoms ne sont pas des variables non plus : ils expriment plutôt la fonction d'identité ($\lambda x.x$).
⇒ Conséquence : plus d'indices en syntaxe (souvent critiqué) et de fonction d'assignation en sémantique.

2.1 Les pronoms : fonctions d'identité

Les pronoms vont dénoter la fonction d'identité ($\lambda x.x$), ce qui suppose quelques modifications dans la sémantique compositionnelle :

(18) *He left.*
$[\![he]\!]$=$\lambda x.x$ → type $\langle e,e\rangle$
$[\![left]\!]$=$\lambda y.y\ left$ → type $\langle e,t\rangle$
$[\![he\ left]\!]$=???

De plus, modifier le type sémantique d'une expression signifie aussi modifier le type syntaxique de cette expression. Les pronoms auront donc le type syntaxique suivant (qui correspond à l'équivalent syntaxique des fonctions naturelles de type $\langle e,e\rangle$) :

(19) $\dfrac{Pronom}{NP^{NP}}$

2.2 Composition de fonction et règle g

En plus de l'application fonctionnelle, possibilité d'utiliser la composition de fonction.

(20) *Composition de fonction :*
Si α et β sont respectivement de type $\langle\sigma,\tau\rangle$ et $\langle a,\sigma\rangle$, alors
$[\![\alpha\beta]\!]$=$[\![\alpha]\!]\circ[\![\beta]\!]$=$\lambda V_a.[\![\alpha]\!]([\![\beta]\!](\mathrm{V}))$.

(21) *He left.*
β= he → type $\langle e,e\rangle$
α= left → type $\langle e,t\rangle$
$[\![he\ left]\!]$=$[\![left]\!]\circ[\![he]\!]$=$\lambda x_e.[\![left]\!]([\![he]\!](x))$ = $\lambda x_e.[\![left]\!](\lambda y.y(x))$=$\lambda x.x\ left$

La Composition de fonction peut parfois être décomposée en Règle g (type-shifting spécial)+Application fonctionnelle.

L'opérateur ou règle g va donc aussi permettre de rendre la combinaison sémantique possible. Cette règle est en fait une règle de *type-shifting*[2] :

(22) *Règle g*
Soit h une fonction de type $\langle a, b\rangle$. Alors g(h) est une fonction de type $\langle\langle c, a\rangle, \langle c, b\rangle\rangle$ telle que $g(h) = \lambda V_{\langle c,a\rangle}[\lambda X_c[h(V(X))]]$

On peut maintenant faire la composition sémantique de (18) en appliquant g à $left$:

(23) $$\begin{aligned}[\![he\ g(left)]\!]&=\lambda f_{\langle e,e\rangle}[\lambda x_e[[\![left]\!](f(x))]](\lambda y.y)\\ &=\lambda x_e.[\![left]\!](\lambda y.y(x))\\ &=\lambda x_e[\lambda v.v\ left](x)\\ &=\lambda x.x\ left\end{aligned}$$

⇒ Le fait d'utiliser la fonction d'identité pour les pronoms fait qu'on obtient une équivalence sémantique entre *left* et *he left*.
⇒ La seule différence, c'est que $left$ n'est pas saturé sémantiquement, alors que $he\ left$ est saturé sémantiquement, mais pas pragmatiquement.
⇒ Si on compare avec la théorie standard, c'est comme si on avait déplacé les entités contextuelles d'un niveau sémantique (la fonction d'assignation avec tous les individus du contexte) vers un niveau supérieur (contexte pragmatique) : $he\ left$ n'aura de valeur de vérité que par l'attribution d'un individu à x.

Problème potentiel ⇒ avec un prédicat à 2 arguments, comment éviter une inversion entre le sujet et l'objet ?

(24) *John called him.*
$$\begin{aligned}[\![John\ g(called)\ him]\!]&=[\lambda f_{\langle e,e\rangle}[\lambda x_e[[\![call]\!](f(x))]](\lambda y.y(x))](John)\\ &=[\lambda x_e.[\![call]\!](\lambda y.y(x))](John)\\ &=[\lambda x_e[\lambda v.\lambda z.z\ called\ v](x)](John)\\ &=[\lambda x.\lambda z.z\ called\ x](John)\\ &=\lambda z.z\ called\ John \quad \text{AU LIEU DE} \quad \lambda x.John\ called\ x.\end{aligned}$$

Solution logique ⇒ le type-shifting sémantique s'associe à un type-shifting syntaxique, d'où une reformulation de la règle g :

(25) Soit α une expression de la forme $\langle[\alpha], B/A, [\![\alpha]\!]\rangle$. Alors, il existe une expression β de la forme $\Big\langle[\alpha],\ \ g_C(B/A) = B^C/A^C,\ \ g_c[\![\alpha]\!] = \lambda V_{\langle c,a\rangle}[\lambda X_c[[\![\alpha]\!](V(X))]]\Big\rangle$ avec A et C des types syntaxiques, et a et c les types sémantiques correspondants.

La dérivation en (24) va être bloquée par les types syntaxiques :

(26)
$$\dfrac{\dfrac{John}{NP} \qquad \dfrac{\dfrac{g(called)}{(S/_L NP)^{NP}/_R NP^{NP}} \qquad \dfrac{him}{NP^{NP}}}{(S/_L NP)^{NP}}}{???}$$

Reste à faire la bonne dérivation. Pour cela il faut utiliser le type-shifting traditionnel puis la règle g sur le sujet $John$:

[2]La différence entre les 2 est la suivante : la composition de fonction est une règle binaire (qui prend 2 entités syntaxiques pour les combiner), alors que la règle g est unaire (modifie le type d'une unité syntaxique pour qu'elle se combine ensuite par application fonctionnelle).

(27) John $= \langle NP, John \rangle$
$\Rightarrow$ l(John) $= \left\langle S/_R(S/_L NP), \lambda P_{\langle e,t\rangle}.P(John) \right\rangle$
$\Rightarrow$ g(l(John)) $= \left\langle S^{NP}/_R(S/_L NP)^{NP}, \lambda R_{\langle e,et\rangle}.\lambda x.[\lambda P_{\langle e,t\rangle}.P(John)](R(x)) \right\rangle$
$= \left\langle S^{NP}/_R(S/_L NP)^{NP}, \lambda R_{\langle e,et\rangle}.\lambda x.R(x)(John) \right\rangle$

Ce qui nous donne la dérivation suivante pour (24) :

(28) Sémantique : $[\![g(l(John))\ g(called)\ him]\!] = [\![g(l(John))]\!]([\![g(called)]\!]([\![him]\!]))$
$= \lambda R_{\langle e,et\rangle}.[\lambda x.R(x)(John)](\lambda f_{\langle e,e\rangle}.[\lambda z_e.[\![call]\!](f(z))]([\![him]\!]))$
$= \lambda R_{\langle e,et\rangle}.[\lambda x.R(x)(John)](\lambda f_{\langle e,e\rangle}.[\lambda z_e.[\![call]\!](f(z))](\lambda y.y))$
$= \lambda R_{\langle e,et\rangle}.[\lambda x.R(x)(John)](\lambda z_e.[\![call]\!](\lambda y.y(z)))$
$= \lambda R_{\langle e,et\rangle}.[\lambda x.R(x)(John)](\lambda z_e.[\![call]\!](z))$
$= \lambda R_{\langle e,et\rangle}.[\lambda x.R(x)(John)](\lambda v.\lambda k.k\ called\ v)$
$= \lambda x.[\lambda v.\lambda k.k\ called\ v](x)(John)$
$= \lambda x.John\ called\ x$

Syntaxe :
$$\dfrac{\dfrac{g(l(John))}{S^{NP}/_R(S/_L NP)^{NP}} \qquad \dfrac{\dfrac{g(called)}{(S/_L NP)^{NP}/_R NP^{NP}} \qquad \dfrac{him}{NP^{NP}}}{(S/_L NP)^{NP}}}{S^{NP}}$$

2.3 Liage local via la règle z

Notez que toute dérivation basée sur la règle g permet seulement d'obtenir une lecture libre des pronoms (renvoyant à un individu du contexte) :

(29) John said that Mary called him.

Syntaxe :
$$\dfrac{\dfrac{g(l(John))}{S^{NP}/_R(S/_L NP)^{NP}} \qquad \dfrac{\dfrac{g(said)}{(S/_L NP)^{NP}/_R S^{NP}} \qquad \dfrac{\dfrac{g(l(Mary)}{S^{NP}/_R(S/_L NP)^{NP}} \qquad \dfrac{\dfrac{g(called)}{(S/_L NP)^{NP}/_R NP^{NP}} \qquad \dfrac{him}{NP^{NP}}}{(S/_L NP)^{NP}}}{S^{NP}}}{(S/_L NP)^{NP}}}{S^{NP}}$$

Sémantique :
$[\![g(l(John)\ g(said)\ g(l(Mary))\ g(called)\ him]\!]$
$= [\![g(l(John))]\!]([\![g(said)]\!]([\![g(l(Mary))\ g(called)\ him]\!]))$
$= [\lambda R_{\langle e,et\rangle}.\lambda x.R(x)(John)]([g([\![said]\!])](\lambda y.Mary\ called\ y))$
$= [\lambda R_{\langle e,et\rangle}.\lambda x.R(x)(John)]([\lambda P_{\langle e,t\rangle}.\lambda z_e.[\![said]\!](P(z))](\lambda y.Mary\ called\ y))$
$= [\lambda R_{\langle e,et\rangle}.\lambda x.R(x)(John)](\lambda z_e.[\![said]\!](\lambda y.Mary\ called\ y(z)))$
$= [\lambda R_{\langle e,et\rangle}.\lambda x.R(x)(John)](\lambda z_e.[\![said]\!](Mary\ called\ z))$
$= [\lambda R_{\langle e,et\rangle}.\lambda x.R(x)(John)](\lambda z_e.[\lambda Q.\lambda v.v\ said\ Q](Mary\ called\ z))$
$= [\lambda R_{\langle e,et\rangle}.\lambda x.R(x)(John)](\lambda z_e.\lambda v.v\ said\ Mary\ called\ z)$
$= \lambda x.[\lambda z_e.\lambda v.v\ said\ Mary\ called\ z](x)(John)$
$= \lambda x.[\lambda v.v\ said\ Mary\ called\ x](John)$
$= \lambda x.John\ said\ Mary\ called\ x$

Mais comment obtenir une lecture anaphorique des pronoms (où $John$ et x renvoient au même individu) ? En utilisant une autre règle (ou autre opérateur), la règle z :

(30) *Règle z*
Soit α une expression de la forme $\langle[\alpha], (A/NP)/B, [\![\alpha]\!]\rangle$. Alors, il existe une expression β de la forme $\left\langle[\alpha],\ \ z((A/NP)/B) = (A/NP)/B^{NP},\ \ z[\![\alpha]\!] = \lambda V_{\langle e,b\rangle}[\lambda x[[\![\alpha]\!](V(x))(x)]]\right\rangle$ avec B un type syntaxique et b le type sémantique correspondant.

D'abord un exemple simple :

(31) John loves his mother

Syntaxe :

$$\dfrac{\dfrac{John}{NP} \qquad \dfrac{\dfrac{z(loves)}{(S/_L NP)/_R NP^{NP}} \qquad \dfrac{his\ mother}{NP^{NP}}}{S/_L NP}}{S}$$

Sémantique :

$[\![John\ z(loves)\ his\ mother]\!]$=$[\![z(loves)]\!]([\![his\ mother]\!])([\![John]\!])$
=$[\lambda f.\lambda x.[\![loves]\!](f(x))(x)](\lambda y.the\ mother\ of\ y)(John)$
=$[\lambda x.[\![loves]\!]([\lambda y.the\ mother\ of\ y](x))(x)](John)$
=$[\lambda x.[\![loves]\!](the\ mother\ of\ x)(x)](John)$
=$[\![loves]\!](the\ mother\ of\ John)(John)$
=$[\lambda v.\lambda k.k\ loves\ v](the\ mother\ of\ John)(John)$
=$John\ loves\ the\ mother\ of\ John$

Retour à nos moutons...

(32) John said that Mary called him(=John).

Syntaxe :

$$\dfrac{\dfrac{John}{NP} \qquad \dfrac{\dfrac{z(said)}{(S/_L NP)/_R S^{NP}} \qquad \dfrac{\dfrac{g(l(Mary)}{S^{NP}/_R (S/_L NP)^{NP}} \qquad \dfrac{\dfrac{g(called)}{(S/_L NP)^{NP}/_R NP^{NP}} \qquad \dfrac{him}{NP^{NP}}}{(S/_L NP)^{NP}}}{S^{NP}}}{S/_L NP}}{S}$$

Sémantique :

$[\![John\ z(said)\ g(l(Mary))\ g(called)\ him]\!]$=$[\![z(said)]\!]([\![g(l(Mary))\ g(called)\ him]\!])([\![John]\!])$
=$[\![z(said)]\!](\lambda y.Mary\ called\ y)(John)$
=$[\lambda P.\lambda x.[\![said]\!](P(x))(x)](\lambda y.Mary\ called\ y)(John)$
=$[\lambda x.[\![said]\!](\lambda y.Mary\ called\ y(x))(x)](John)$
=$[\lambda x.[\![said]\!](Mary\ called\ x)(x)](John)$
=$[\![said]\!](Mary\ called\ John)(John)$
=$[\lambda Q.\lambda v.v\ said\ Q](Mary\ called\ John)(John)$
=$John\ said\ Mary\ called\ John.$

3 Tout ça pour quoi...

3.1 Questions fonctionnelles

Cett théorie permet d'expliquer les lectures fonctionnelles obtenues dans les exemples suivants :

(33) (a) Qui est-ce que chaque homme aime ? Sa mère.
(b) Quelle femme est-ce qu'aucun homme n'oublierait d'inviter à son mariage ? Sa mère.

Selon Engdahl (1980), ces questions doivent être interprétées comme fonctionnelles :

(34) (a) Quelle est la fonction f telle que chaque homme x aime $f(x)$
(b) Quelle est la fonction f (sur l'ensemble des femmes) telle qu'aucun homme x n'oublierait d'inviter $f(x)$ au mariage de x.

Ce sont donc des questions sur des fonctions de type $\langle e, e\rangle$. Pour en rendre compte, Engdahl (1980) stipule que les traces peuvent être fonctionnelles. En sémantique sans variable, cette stipulation est inutile (même les traces sont inutiles !), comme le montre le calcul de (33a) par composition de fonction :

(35) β= z(aime) → type $\langle ee, et\rangle$
α= chaque homme → type $\langle et, t\rangle$
$[\![chaque\ homme\ z(aime)]\!]=[\![chaque\ homme]\!]\circ[\![z(aime)]\!]$
$=\lambda f_{\langle e,e\rangle}.[\![chaque\ homme]\!]([\![z(aime)]\!](f))$
$=\lambda f_{\langle e,e\rangle}.[\![chaque\ homme]\!]([\lambda h.\lambda x.[\![aime]\!](h(x))(x)](f))$
$=\lambda f_{\langle e,e\rangle}.[\![chaque\ homme]\!](\lambda x.[\![aime]\!](f(x))(x))$
$=\lambda f_{\langle e,e\rangle}.[\lambda P.\forall y.homme(y) \rightarrow P(y)](\lambda x.[\![aime]\!](f(x))(x))$
$=\lambda f_{\langle e,e\rangle}.[\forall y.homme(y) \rightarrow [\lambda x.[\![aime]\!](f(x))(x)](y)]$
$=\lambda f_{\langle e,e\rangle}.[\forall y.homme(y) \rightarrow [\![aime]\!](f(y))(y)]$
$=[\lambda f.[\forall y.homme(y) \rightarrow [\lambda x.\lambda v.v\ aime\ x](f(y))(y)]]$
$=[\lambda f.[\forall y.homme(y) \rightarrow [\lambda v.v\ aime\ f(y)](y)]]$
$=[\lambda f.[\forall y.homme(y) \rightarrow y\ aime\ f(y)]]$

Notez ici qu'il est toujours possible de décomposer la composition de fonction en règle g+application fonctionnelle, mais la combinaison syntaxique serait impossible :

(36) $[\![chaque\ homme\ z(aime)]\!]=[\![g_{\langle ee\rangle}(chaque\ homme)]\!]([\![z(aime)]\!])$
$=[\lambda R_{\langle ee,et\rangle}.\lambda f_{\langle ee\rangle}.[\![chaque\ homme]\!](R(f))]([\![z(aime)]\!])$
$=[\lambda R_{\langle ee,et\rangle}.\lambda f.[\lambda P.\forall y.homme(y) \rightarrow P(y)](R(f))]([\![z(aime)]\!])$
$=[\lambda R_{\langle ee,et\rangle}.\lambda f.[\forall y.homme(y) \rightarrow R(f)(y)]]([\![z(aime)]\!])$
$=[\lambda f.[\forall y.homme(y) \rightarrow [\![z(aime)]\!](f)(y)]]$
$=[\lambda f.[\forall y.homme(y) \rightarrow [\lambda h.\lambda x.[\![aime]\!](h(x))(x)](f)(y)]]$
$=[\lambda f.[\forall y.homme(y) \rightarrow [\lambda x.[\![aime]\!](f(x))(x)](y)]]$
$=[\lambda f.[\forall y.homme(y) \rightarrow [\![aime]\!](f(y))(y)]]$
$=[\lambda f.[\forall y.homme(y) \rightarrow [\lambda x.\lambda v.v\ aime\ x](f(y))(y)]]$
$=[\lambda f.[\forall y.homme(y) \rightarrow [\lambda v.v\ aime\ f(y)](y)]]$
$=[\lambda f.[\forall y.homme(y) \rightarrow y\ aime\ f(y)]]$

⇒ La règle g sur *chaque homme* modifierait le type syntaxique $(S^{NP^{NP}}/_R(S/_L NP)^{NP^{NP}})$ et le rendrait incompatible avec celui de *z(aime)* $((S/_L NP)/_R NP^{NP})$.

⇒ Plus besoin de traces (encore moins de traces fonctionnelles) pour expliquer les questions fonctionnelles, seulement le sens des unités linguistiques et les règles de composition (composition de fonction et z).

3.2 Phrases équationnelles

Cette théorie permet une analyse similaire des lectures fonctionnelles dans les phrases équationnelles :

(37) La femme que chaque homme aime est sa mère.

Comment expliquer l'interprétation de variable liée de *sa* ? En sémantique sans variable, ceci revient à une équivalence entre 2 fonctions :

(38) $\iota f_{\langle ee\rangle}$(dans le domaine des femmes).$[\![chaque\ homme]\!]([\![z(aime)]\!](f)) = \lambda x.la\ mère\ de\ x$
$\iota f_{\langle ee\rangle}$(dans le domaine des femmes).$[\forall y.homme(y) \rightarrow y\ aime\ f(y)] = \lambda x.la\ mère\ de\ x$

3.3 Le croisement

Les faits de croisement faible découlent automatiquement du calcul sémantique sans variable.

(39) (a) Chaque homme$_1$ aime sa$_1$ mère.
(b) *Sa$_1$ mère aime chaque homme$_1$.

Seule la règle z donne la lecture de variable liée, mais le pronom ne peut être lié que par un argument plus haut (l'argument externe x va lier un élément de l'argument interne $f(x)$. Pour obtenir le liage en (b), il faudrait une règle inverse de z du type $z_{inverse}[\![\alpha]\!]= \lambda V_{\langle e,b\rangle}.[\lambda x[[\![\alpha]\!](x)(V(x))]]$. Mais cette règle n'existe pas.

3.4 Le liage Across-The-Board

Concerne les constructions à Right-Node-Raising (extraposition à droite)

(40) chaque homme aime et aucun homme n'épouse sa mère.

⇒ Contrainte sur les lectures possibles : si le pronom *sa* est lié dans un conjoint, il doit aussi l'être dans l'autre (sorte de contrainte de parallélisme à la Fox (2000)).

Cette contrainte suit directement du calcul sémantique sans variable. La composition sera possible seulement si les deux conjoints utilisent la même règle (soit z pour la lecture liée, soit g pour la lecture libre).

(41) $[\![chaque\ homme]\!] \circ [\![z(aime)]\!]$; $\mathrm{S}/_R\mathrm{NP}^{NP}$;
$\lambda f.[\forall x.homme(x) \rightarrow x\ aime\ f(x)]$
$[\![aucun\ homme]\!] \circ [\![z(épouse)]\!]$; $\mathrm{S}/_R\mathrm{NP}^{NP}$;
$\lambda g.[\neg\exists y.homme(y) \rightarrow y\ épouse\ g(y)]$
$[\![chaque\ homme\ aime\ et\ aucun\ homme\ n'épouse]\!]$
$= \lambda f.[\forall x.homme(x) \rightarrow x\ aime\ f(x)] \sqcap \lambda g.[\neg\exists y.homme(y) \rightarrow y\ épouse\ g(y)]$
$= \lambda f.[\forall x.homme(x) \rightarrow x\ aime\ f(x) \sqcap \neg\exists y.homme(y) \rightarrow y\ épouse\ f(y)]$

(42) $[\![chaque\ homme]\!] \circ [\![z(aime)]\!]$; $\mathrm{S}/_R\mathrm{NP}^{NP}$;
$\lambda f.[\forall x.homme(x) \rightarrow x\ aime\ f(x)]$
$[\![aucun\ homme]\!] \circ [\![épouse]\!]$; $\mathrm{S}/_R\mathrm{NP}$;
$\lambda x.[\neg\exists y.homme(y) \rightarrow y\ épouse\ x]$
$[\![et]\!]=(\mathrm{X}/_L\mathrm{X})/_R\mathrm{X}$; $\lambda A.\lambda B.[B \sqcap A]$

⇒ La conjonction doit prendre 2 arguments de même type (syntaxique et sémantique), ce qui n'est pas possible en (42).

3.5 Anaphores de type e

La sémantique sans variable peut aussi rendre compte des lectures de type e des pronoms, et ceci sans stipuler une variable cachée pour obtenir la lecture fonctionnelle :

(43) Chaque homme$_1$ aime sa$_1$ mère, et aucun homme$_2$ ne l'épouse.

Le pronom *l'* peut renvoyer à une personne différente pour chaque homme (i.e. sa mère).
⇒ Sous cette lecture, il ne peut être interprété comme une variable libre d'individu (lecture individuelle).

⇒ L'interprétation de variable liée est elle aussi impossible (le pronom renvoyant à une mère, pas à chaque homme).

Cette lecture découle très simplement du calcul sémantique sans variable en appliquant la règle g_e au pronom (type-shifting) :

(44) $[\![g(la)]\!]=\lambda f_{ee}.\lambda x.[\![la]\!](f(x)) = \lambda f_{ee}.\lambda x.[\lambda y.y](f(x)) = \lambda f_{ee}.\lambda x.[f(x)] = \lambda f.f$
(type $\langle ee, ee\rangle$)
$[\![g_{ee}(z(\textit{épouse}))]\!] = \lambda D.[\lambda h.\lambda x.[\![\textit{épouse}]\!](D(h)(x))(x)]$
(type $\langle\langle ee, ee\rangle, \langle ee, et\rangle\rangle$)
$[\![g_{ee}(\textit{aucun homme})]\!]=[\lambda R_{\langle ee,et\rangle}.\lambda f_{\langle ee\rangle}.[\![\textit{aucun homme}]\!](R(f))]$
$=[\lambda R_{\langle ee,et\rangle}.\lambda f.[\lambda P.\neg\exists y.\textit{homme}(y) \rightarrow P(y)](R(f))]$
$=[\lambda R_{\langle ee,et\rangle}.\lambda f.[\neg\exists y.\textit{homme}(y) \rightarrow R(f)(y)]]$
(type $\langle\langle ee, et\rangle, \langle ee, t\rangle\rangle$)
$[\![$g(aucun homme) g(z(épouse)) g(la)$]\!] = \lambda f.[\neg\exists x.\textit{homme}(x) \rightarrow x\ \textit{épouse}\ f(x)]$

⇒ Cette proposition n'aura de valeur de vérité que par l'attribution d'une valeur à f venant du contexte. Ici, le contexte fournit clairement la fonction naturelle *mère-de* de type $\langle ee\rangle$

4 Quelques problèmes

4.1 Contexte approprié ?

Qu'est-ce qui peut satisfaire cette notion de contexte approprié ? Par exemple, pour le cas des pronoms de type e, comment expliquer le contraste suivant :

(45) (a) Chaque homme$_1$ aime sa$_1$ mère. Aucun homme$_2$ ne l'épouse.
$\forall x.\textit{homme}(x) \rightarrow \textit{aime la mère de } x$ $\lambda f.[\neg\exists y.\textit{homme}(y).y\ \textit{épouse}\ f(y)$
(b) Chaque homme aime la mère de Marie. Aucun homme ne l'épouse.
$\forall x.\textit{homme}(x) \rightarrow \textit{aime la mère de Jean}$ $\lambda f.[\neg\exists y.\textit{homme}(y).y\ \textit{épouse}\ f(y)$

⇒ Seul (a) permet une lecture distributive du pronom (*l'* pouvant renvoyer à une mère différente), (b) ne peut avoir cette lecture. Pourtant (b) fournit aussi dans son contexte une fonction naturelle, i.e. la fonction *mère-de*. Autrement dit, en quoi la présence d'un pronom dans la fonction naturelle devrait-elle faire une différence ?

4.2 Le cas des pronoms résomptifs

Bien difficile de voir comment Jacobson (1999) pourrait traiter le contraste suivant :

(46) (a) *Quelle femme$_1$ chaque homme a-t-il embrassée $\_\__1$?* *(SA/FA)*
(b) *Quelle femme$_1$ es-tu fâché parce que chaque homme* **l**$_1$ *'a embrassée ?* *(SA/*FA)*

⇒ Les lectures fonctionnelles disparaissent avec un pronom résomptif. Pourtant, la sémantique sans variable ne prédit aucune différence entre lacune et pronom ($\lambda x.x$) : les 2 devraient pouvoir légitimer une lecture fonctionnelle.

Références

Elisabet Engdahl. *The Syntax and Semantics of Questions in Swedish.* PhD thesis, University of Amherst, MA, 1980.

Danny Fox. *Economy and the semantic interpretation.* MIT Press, 2000.

Irene Heim and Angelika Kratzer. *Semantics in generative grammar.* Blackwell, 1998.

Pauline Jacobson. Towards a variable-free semantics. *Linguistics and Philosophy*, 22 :117–184, 1999.